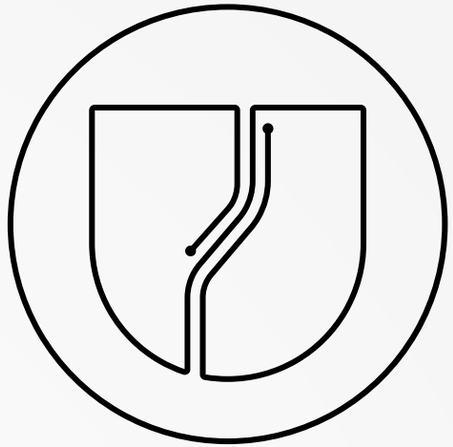
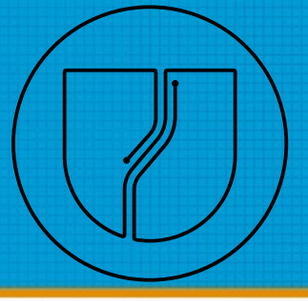
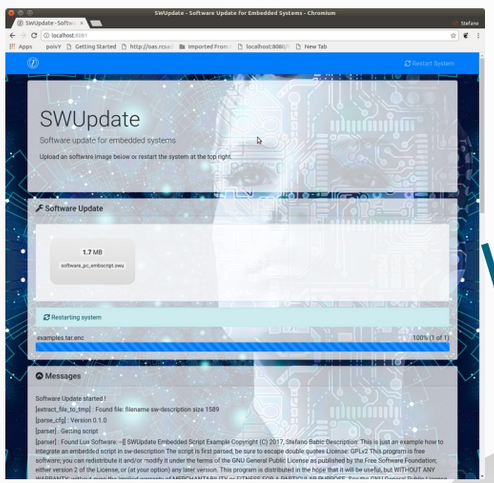


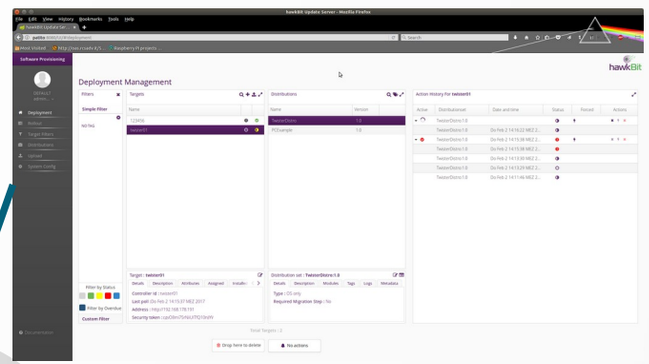
Delta OTA Update with SWUpΔate



- Which is a Δ (incremental) update
- FOSS projects for Δ
- Δ and SWUpdate

SWUpdate: OTA Update Agent

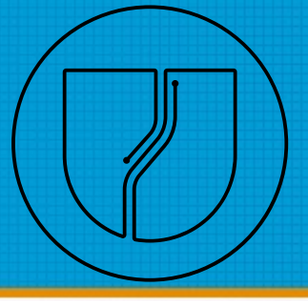




Network

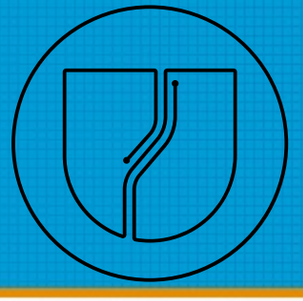
https://swupdate.org

Reasons for Δ updates



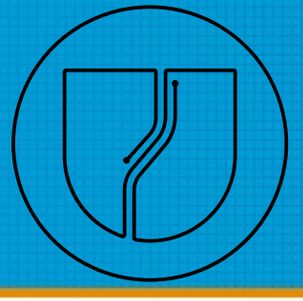
- Size of SW is increasing
- Bandwidth constraints (GSM,etc.)
- Cost for device owner / hosting server

Ways in SWUpdate



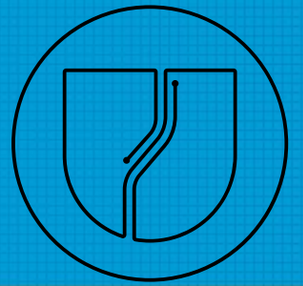
- Split OS and application
  - Update of application is smaller
  - Consistency ??
- Delta update based on librsync

Split OS + App



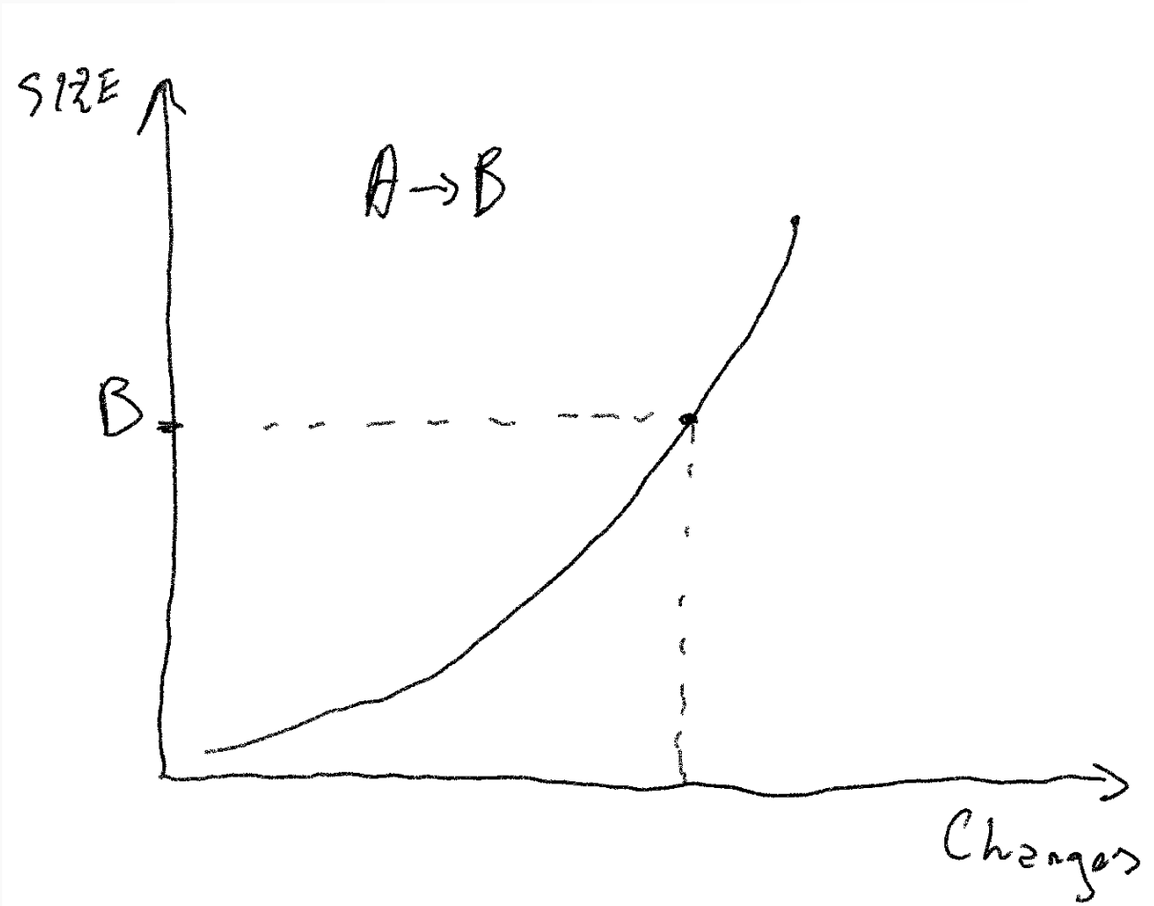
```
appimage :(
        {
          filename = "myapp.tgz";
          type = "archive";
          device = "dev to be used";
          path = "/";
          filesystem = "ext4";
          sha256 =<computed hash>;
        }
);
```

Use librsync handler

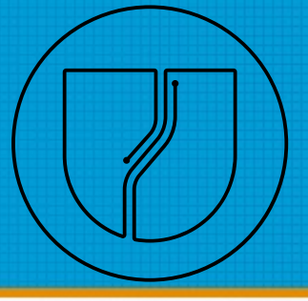


- Prepare delta during build
- Δ File for each source version

```
images: (
    {
        type = "rdiff_image";
        filename = "image.rdiff.delta";
        device = "/dev/mmcblk0p2";
        properties: {
            rdiffbase = ["/dev/mmcblk0p1"];
        };
    });
```

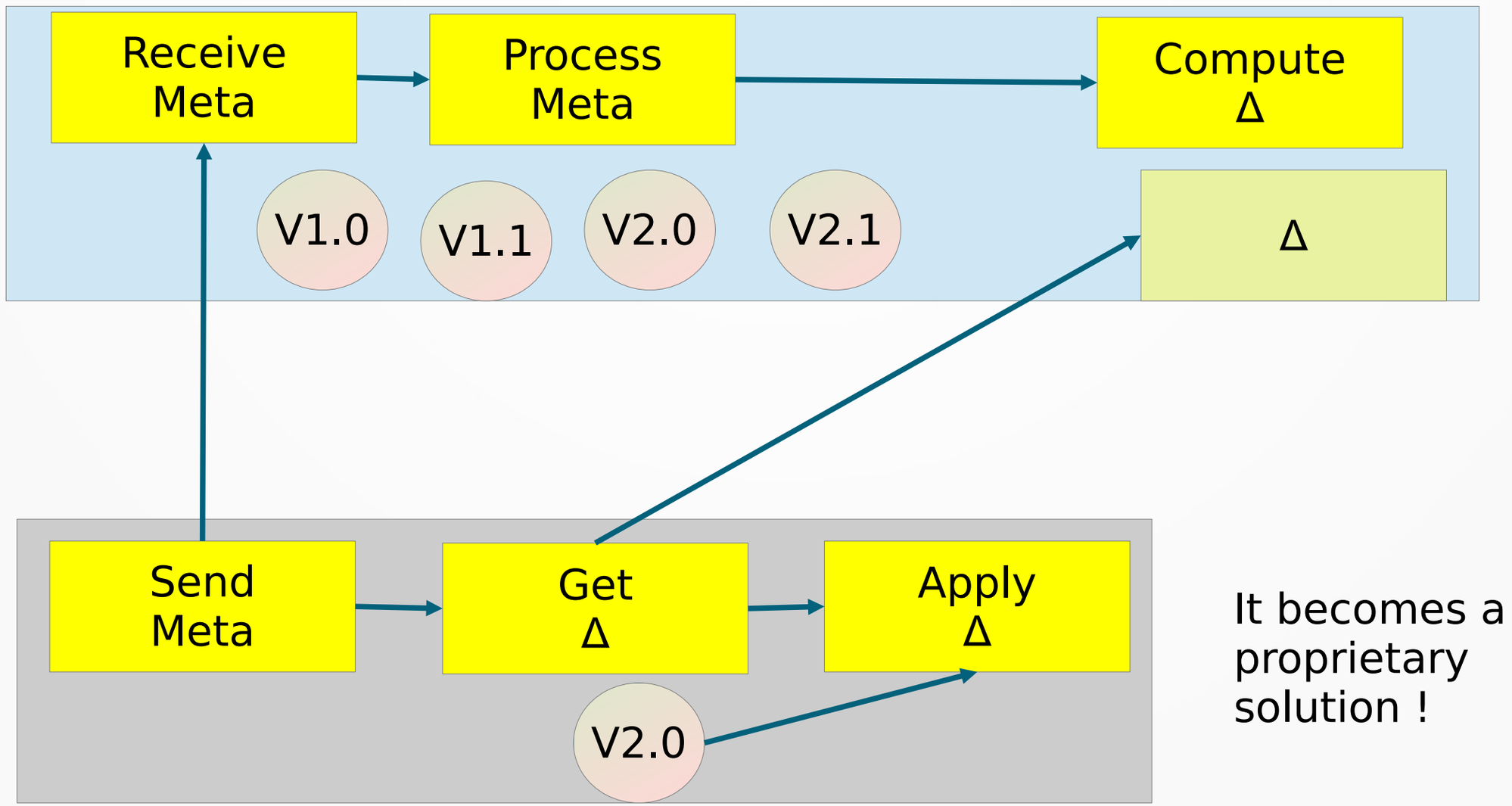


- Rdiff → Δ
- OK rolling releases
- Prebuilt Δ images
- No suitable for update from any release to any release



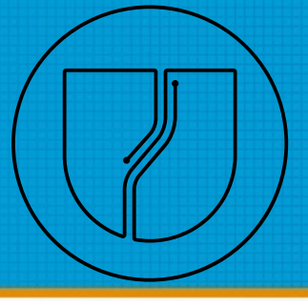
- Independent from type of sources
  - Δ is smaller if type is known
- Low Resources to create the destination
- Δ + SRC → DST(device) == DST(Build)

Deployment Server



It becomes a proprietary solution !

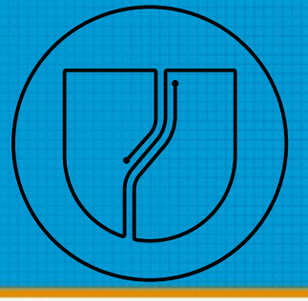
Different concept



- Δ on Server
  - Server holds all versions (old and new)
  - Generates multiple Δ files
  - Check based just on info sent by device
  + Less CPU Load on device
  + Update faster

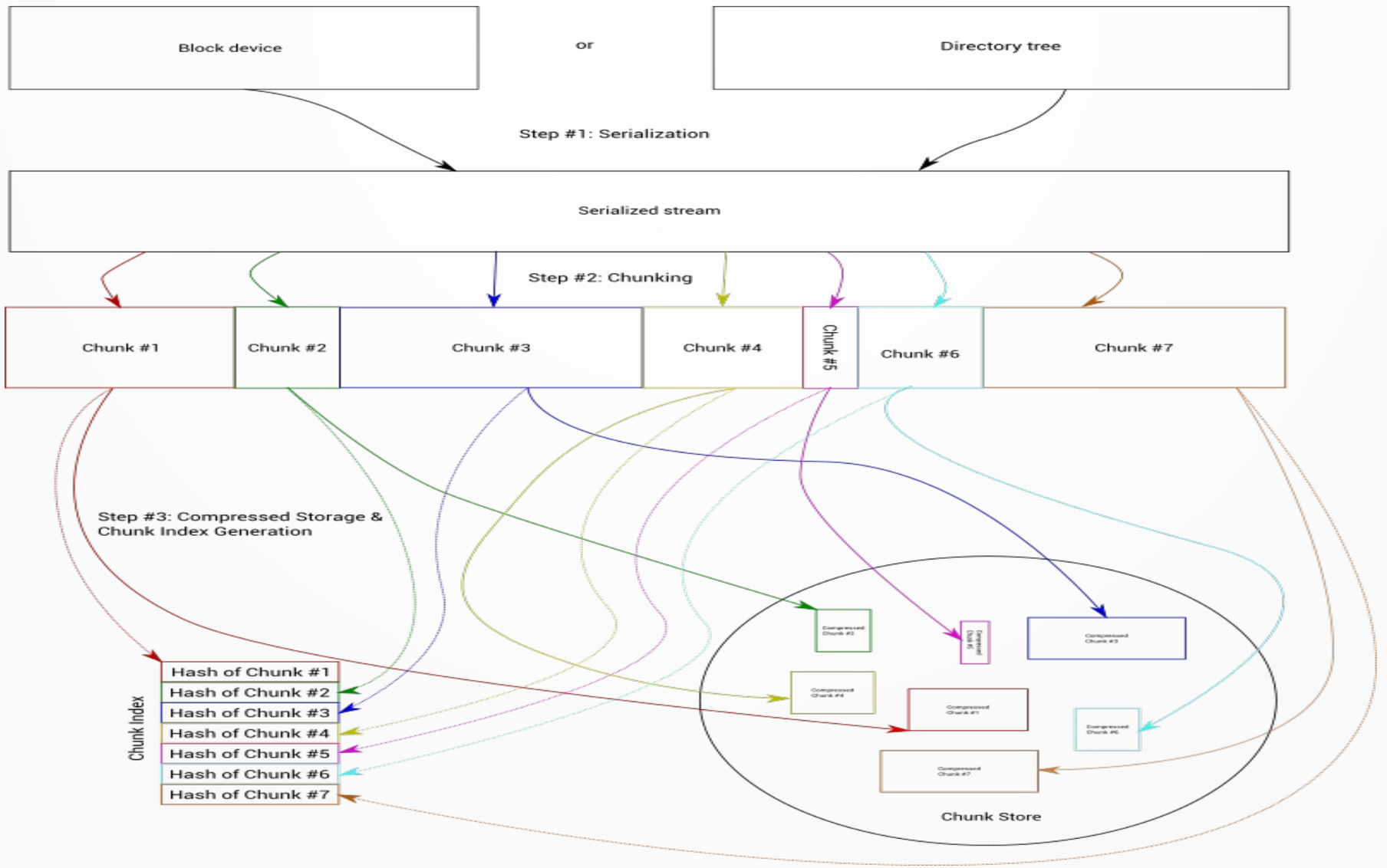
- Δ on Device
  + Server holds one version
  + Δ on device → same server
  + Δ from any X to any Y
  + Crypto check
  - More CPU Load on device
  - Update slower

FOSS to build Δ



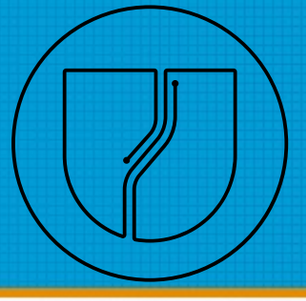
- X-delta
  - Resulting image is built in RAM
  - Not suitable for embedded
  - Δ (like librsync) specific for each version
- Librsync → already treated
- casync

casync



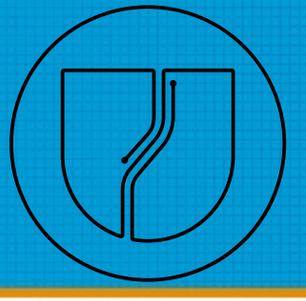
https://0pointer.net/blog/casync-a-tool-for-distributing-file-system-images.html

Casync with SWUpdate ?



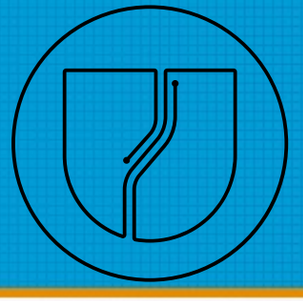
- Each chunk is a separate file
  - Device must download hundreds of separate files
  - Sometimes FW stored by another entity
  - HTTP(S) requests are expensive on small devices.
- It is a complex project
  - Integration difficult.
  - Library was planned, never implemented.
  - Breaks security („privilege separation“)

Zchunk



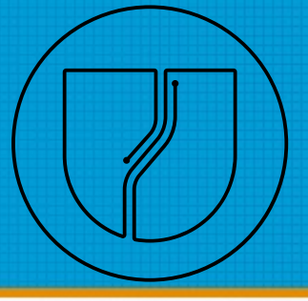
- Developed by Jonathan Dieter
  - https://github.com/zchunk/zchunk
- Used on Fedora Project
- Define a new file format
  - Self contained
  - Meta (index) is part of delivered file

Affinity with Zchunk



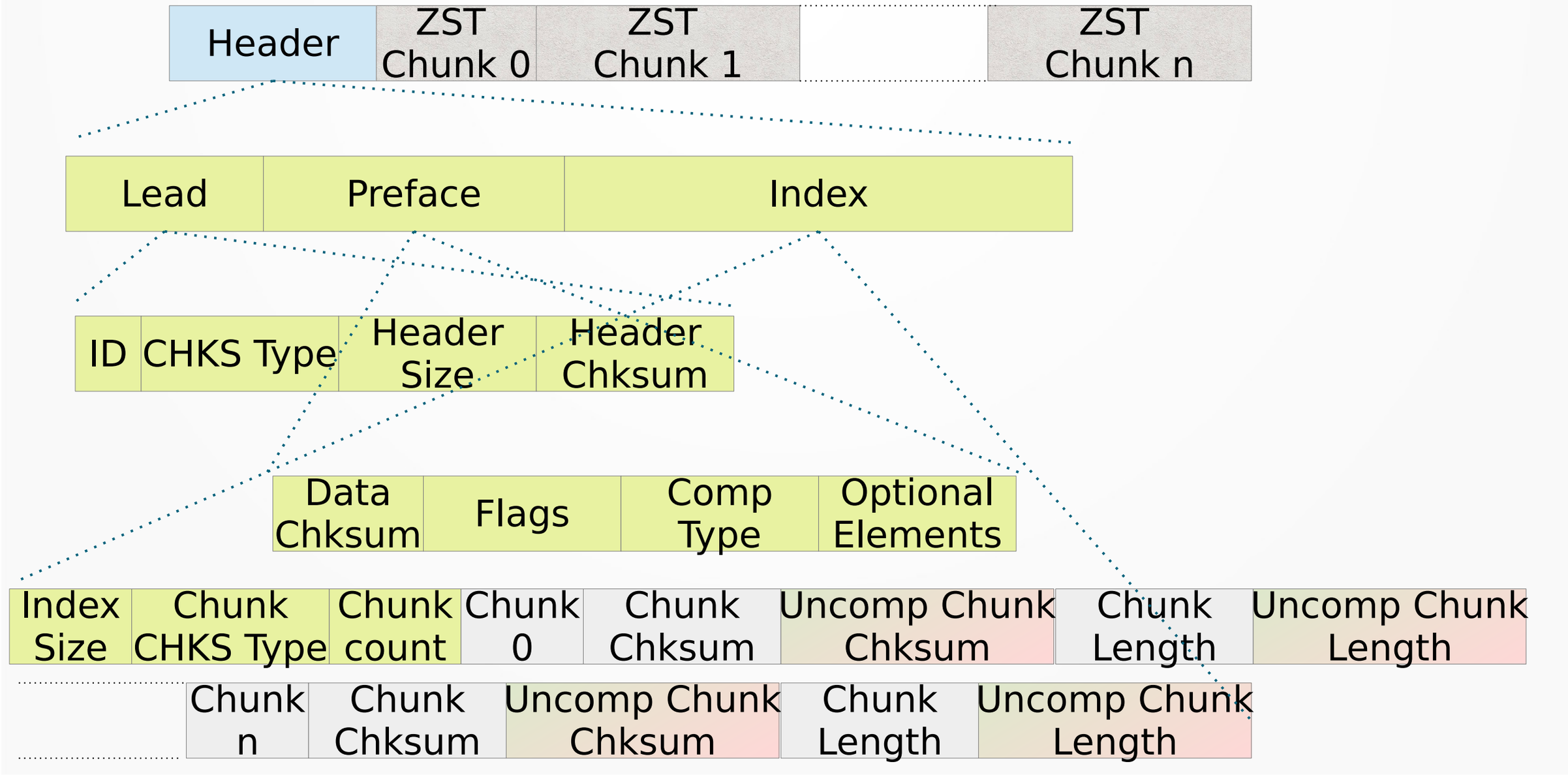
The more I looked at casync, the more obvious it became that it’s designed for a different use-case (delivering full filesystem images), and, while close, wasn’t quite what I needed.

Changes in Zchunk

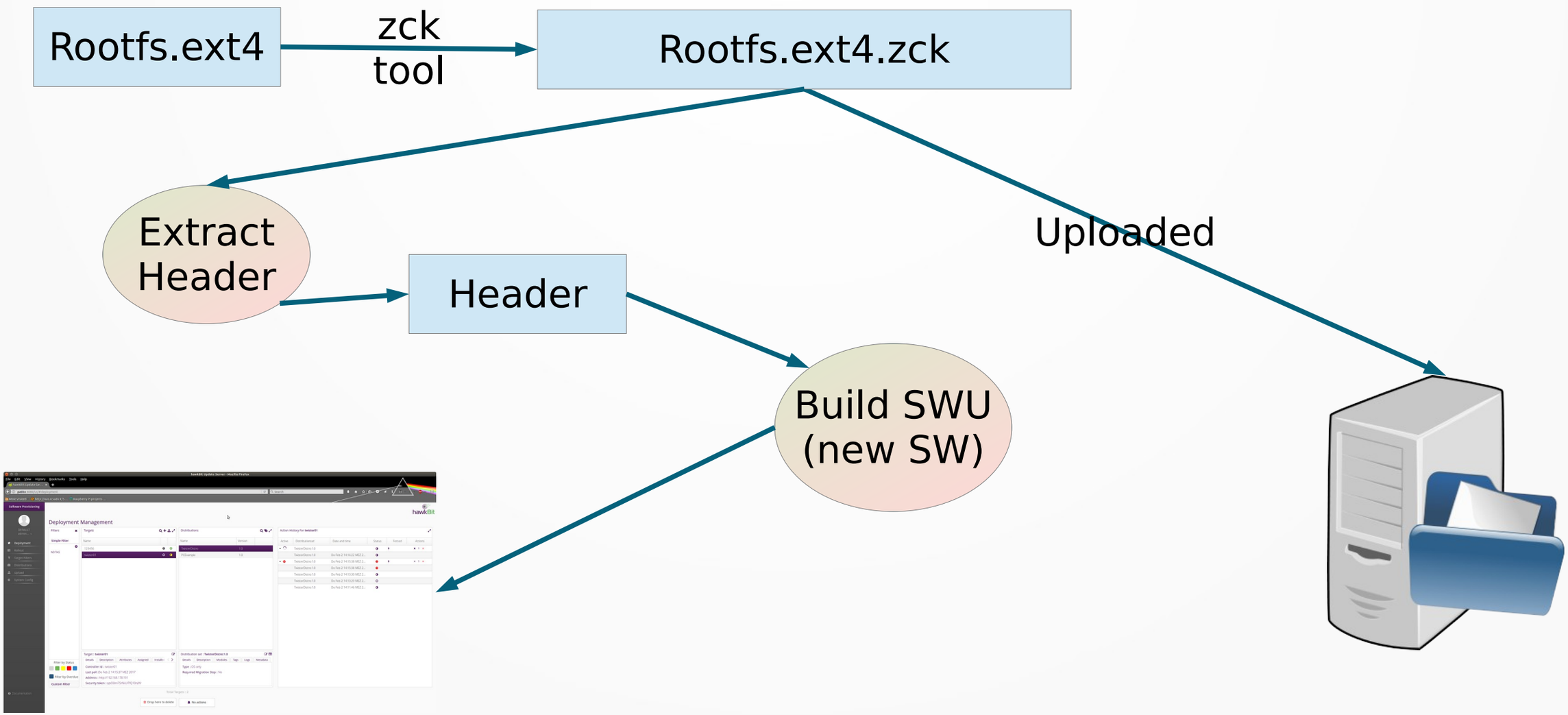


- Embedded friendly
  - Runtime errors instead assert / exit
  - No load file in memory to build ZCK
- Add uncompressed Hashing
- Extend API
  - To build index
  - To return list of chunks meta for old and new version
- Extend format (fields for uncompressed, etc.)
- Merged by Jonathan since 1.2.0

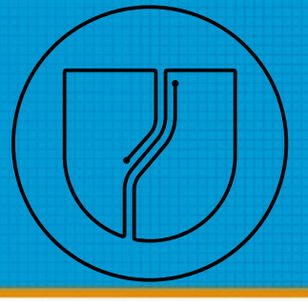
Zchunk format



Build


Rootfs.ext4
zck
tool
Rootfs.ext4.zck
Extract
Header
Header
Uploaded
Build SWU
(new SW)

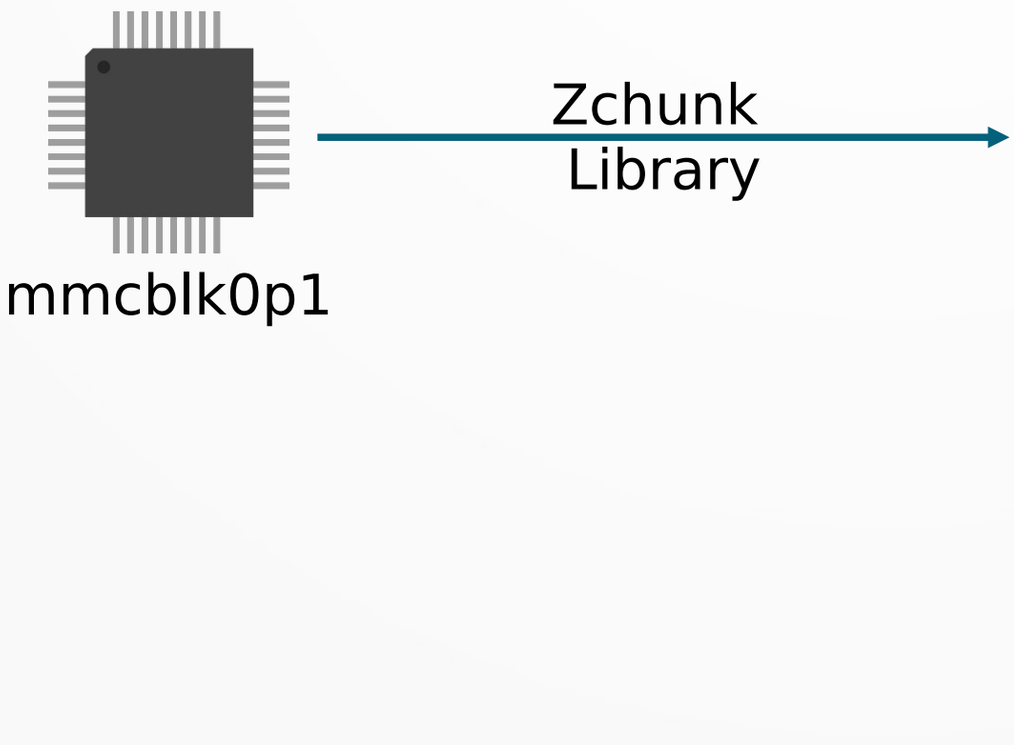
Delta in SWU



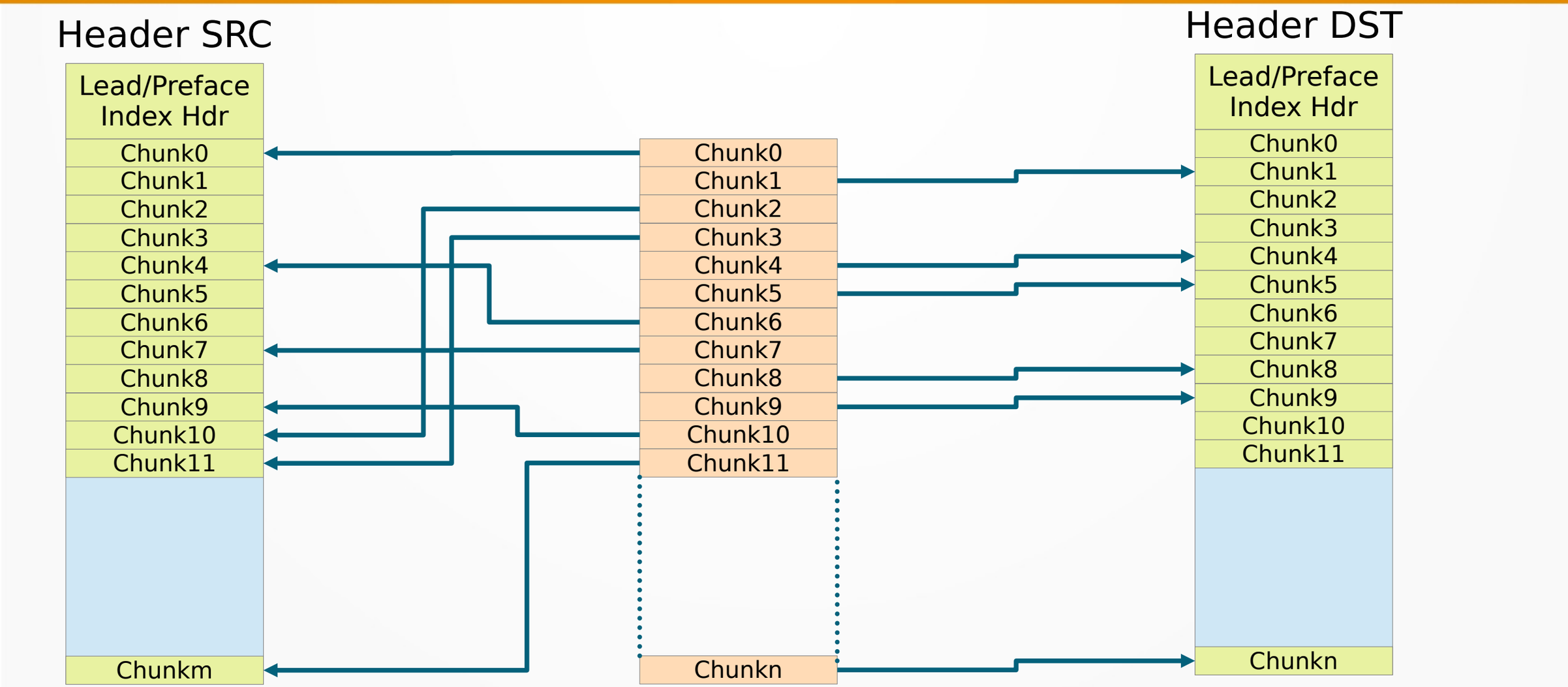
```
images: (
   {
      filename = "myimage.rootfs.ext4.zck.header";
      type = "delta";
      device = "/dev/mmcblk0p2";
      properties: {
         url = "https://examples.com/my.rootfs.ext4.zck";
         chain = "raw";
         source = "/dev/mmcblk0p1";
         zckloglevel = "error";
         debug-chunks = "true";
      };
   }
```

The destination

Analyze source



Look up with UT Hash

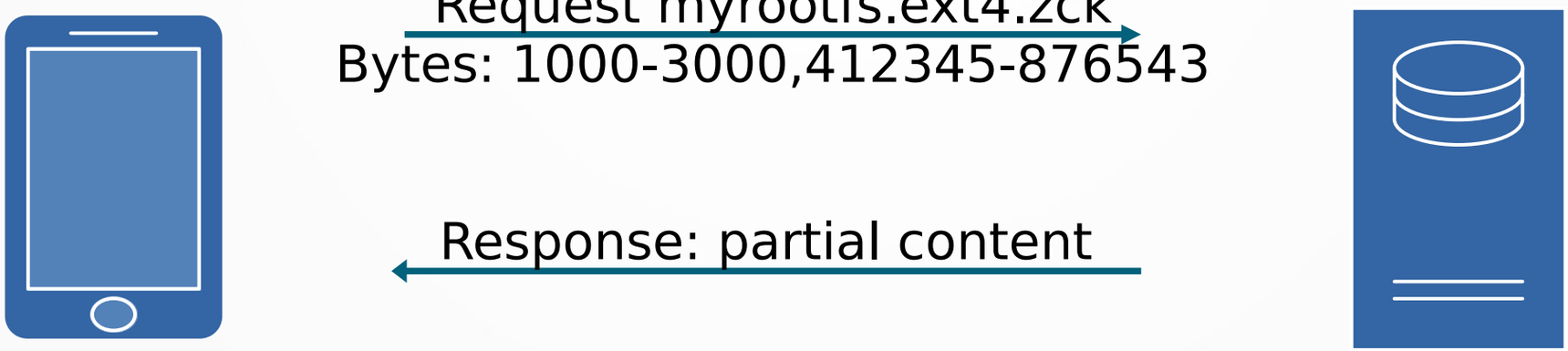


Generated

Downloaded in SWU

https://troydhanson.github.io/uthash/

RFC 7233 – Range request

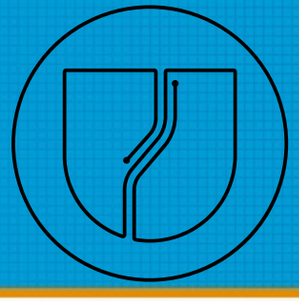


Multiple chuncks in a single HTTP GET

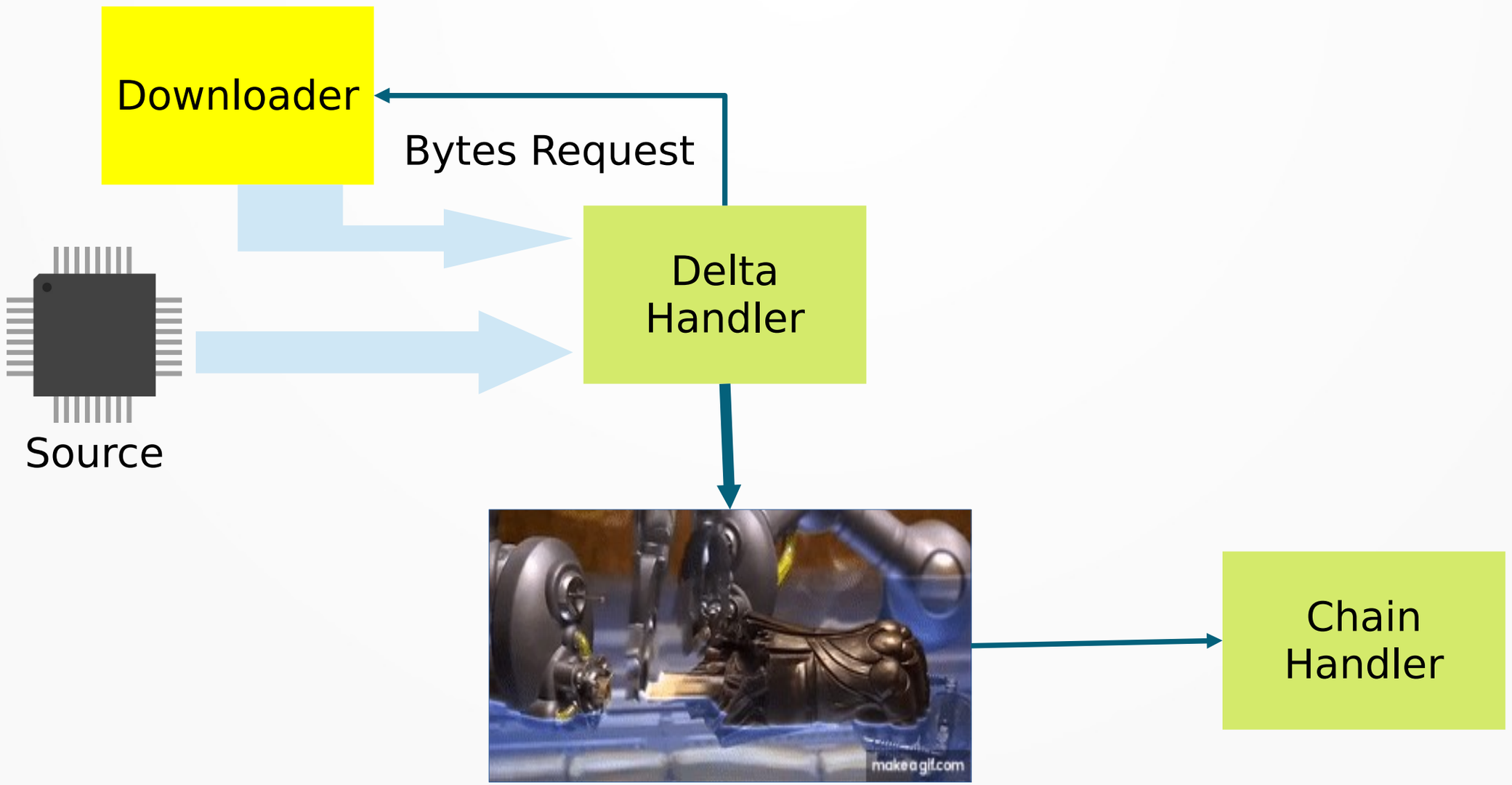
Servers have a max range number

- Device must queue multiple requests

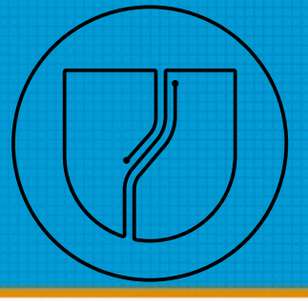
Download chunks



```
GET /rootfs.ext4.zck HTTP/1.1
Host: example.com
Range: bytes=24568-345678,435678-980123

HTTP/1.1 206 Partial Content
Content-Type: multipart/byteranges; boundary=ab1234cde5678
Content-Length: 321110

--ab1234cde5678
Content-Type: bytes
Content-Range: bytes 24568-345678

      → binary data
--ab1234cde5678
.................
```

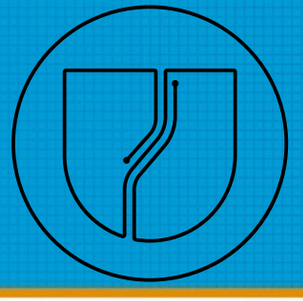

Downloader
Bytes Request
Delta Handler
Source
makeagif.com
Chain Handler

Extract from LOG



```
[TRACE] : [get_total_size] :   Index     Typ HASH                                                                 START(chunk) SIZE(uncomp) Pos(Device)
SIZE(comp)
[TRACE] : [get_total_size] :        42 SRC 6e7f552dad1491de3d6b8a0e4cdb9de3f0bf8cb887079e0cfaff29671eb104fe     679747       9568     2709834        504
[TRACE] : [get_total_size] :        43 DST 22e4804633181c025ae3a0b783cf8f55d668bc563d439fe18962a88d892c464f     680251     117672     2719402       5156
[TRACE] : [get_total_size] :        44 DST f276c49bda860b0272e0d7545b9abd8efad722f41469873f6e619388663c73f5     685407       8970     2837074        469
[TRACE] : [get_total_size] :        45 DST dfd97188ec2fbf14660781ca64b9da516d23de466681fe0440a07a71f12029d6     685876     131072     2846044       5477
[TRACE] : [get_total_size] :        46 DST be2deecb250ae31818dae3c1e31b5f225e2d71c6a895f03da9b0e2dc151ac8f5     691353     131072     2977116       5670
[TRACE] : [get_total_size] :        47 DST 7beda48b777329c02bc260a89a8d0ce71b2e9f9c7d298b6d1327f956e9c315f9     697023      24640     3108188       1153
[TRACE] : [get_total_size] :        48 DST d61c717d21c64cb8798feb7134f0cd4bf61a107d9f4e0188effa38bbe92fd439     698176       9421     3132828        502
[TRACE] : [get_total_size] :        49 DST b2b09feb6c73526c22c5c945ef0b44b8cb7a5f6b5a5022399fd637b366012a05     698678     131072     3142249       5625
[TRACE] : [get_total_size] :        50 DST 87f8d7c921a9afb0ad510f0f03fcd2158f308f8356e59f27510415f5a84b49d9     704303      20678     3273321       1032
[TRACE] : [get_total_size] :        51 DST 7d389d45a011832dbc9188d84c61393b59691b526484f6cdfc3e193759befcf5     705335     131072     3293999       5853
[TRACE] : [get_total_size] :        52 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711188     131072     3425071         22
[TRACE] : [get_total_size] :        53 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711210     131072     3556143         22
[TRACE] : [get_total_size] :        54 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711232     131072     3687215         22
[TRACE] : [get_total_size] :        55 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711254     131072     3818287         22
[TRACE] : [get_total_size] :        56 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711276     131072     3949359         22
[TRACE] : [get_total_size] :        57 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711298     131072     4080431         22
[TRACE] : [get_total_size] :        58 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711320     131072     4211503         22
[TRACE] : [get_total_size] :        59 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711342     131072     4342575         22
[TRACE] : [get_total_size] :        60 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471     711364     131072     4473647         22

[INFO ] : [get_total_size] : Total bytes to be reused     :      637800925

[INFO ] : [get_total_size] : Total bytes to be downloaded :       197733

[INFO ] : [install_delta] : Size of artifact to be installed :    641728512
```

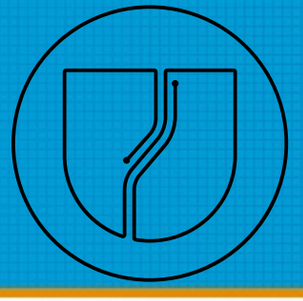
Bandwidth at build time



- Generate ZCK
  - zck -u -h sha256 -v -o rootfs.ext4.zck rootfs.ext4
- Tool to get difference (exposed as test in Zchunk)
  - zck_cmp_uncomp rootfs.ext4.old rootfs.ext4.zck

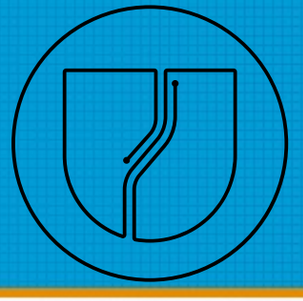
```
8107 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540064      131072
8108 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540086      131072
8109 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540108      131072
8110 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540130      131072
8111 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540152      131072
8112 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540174      131072
8113 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540196      131072
8114 SRC fa43239bcee7b97ca62f007cc68487560a39e19f74f3dde7486db3f98df8e471    83540218      131072
8115 DST d69ae673b9891676f11ece248286fc0eb773e1ae799c1869aabf9f74ca744805    83540240      122229

Total to be reused :          602976803
Total to be downloaded :      9008805
```

Todo



- It was commissioned for Debian / ISAR
  - Missing support in meta-swupdate
  - Methods to generate .zck
  - Build of SWU
- Optimizations ?

Contributions



- Δ handler with dual-copy update.
- No changes on Server side
- Just store the ZCK file
- Bytes to be downloaded known at build time